Capstone Project 1

Predicting demand for cycle hire scheme

Author: Vanita Kalaichelvan

January 14, 2020

Contents

Chapter 1

Introduction

1.1 Problem Statement

The pubic bike hire scheme is a great way of maximising the efficiency of public transportation systems while reducing pollution. In London, the Sanatander cycle hire scheme was started in 2010 (originally sponsored by Barclays) and it continues to be an envionmentaly friendly and healthy way to explore the city or just commute. Another useful aspect of the scheme is that bikes can be hired at any time of the day and there are 839 stations densely populated around central London.

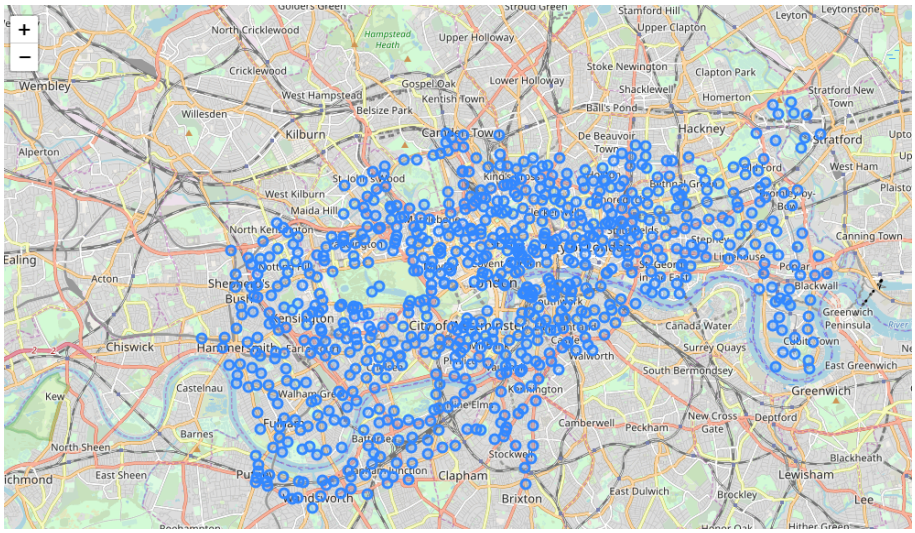

Figure 1.1: Location of cycle stations

In recent years, more players are entering the cycle hire scheme such as Mobike and Ofo that offer lower pricing for short trips under 30 minutes with a more seamless experience that doesn't require docking. While we cannot change the way the bikes are tracked, we can improve overall output of the system by matching availability with demand and identifying under utilized or over utilized spaces. In a recent survey done about the public cycle hire scheme, the key consumer concerns were bike availability and space availability at docking stations (see quarterly reports released by the scheme)

In this project, we will use historical data provided by the trip data to build a model that will predict the upcoming day's ride count and average duration based on historical observations. This is known as time series forecasting.

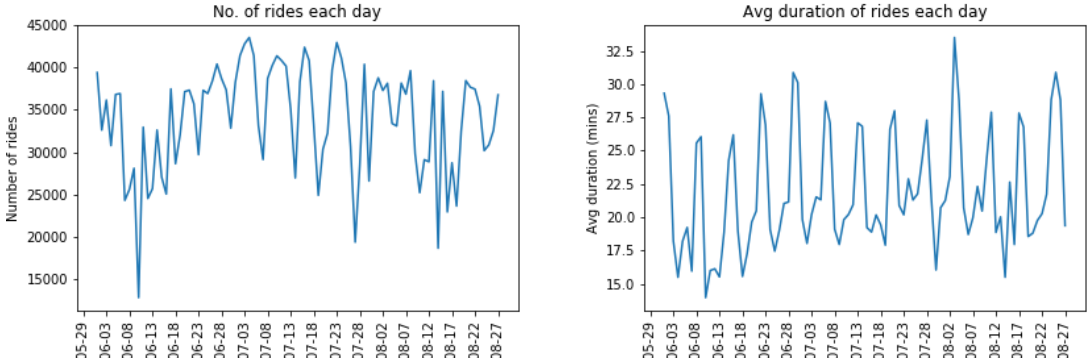

Figure 1.2: Number of rides and average duration per day

1.2 Description of dataset

The dataset is provided by TFL (Transport for London) which has a record of every trip made. An example of the dataset can be seen in Figure 1.3. You can also refer to Part 1 of the Jupyter notebooks, which shows how the data is downloaded from the AWS file storage system, S3. The easiest way is to use `boto 3` to access and read each file into a string, concatenate the strings into a large string object and then, write the object to a `pandas` dataframe. This helps to save memory space and improves time performance.

	Rental Id	Duration	Bike Id	End Date	EndStation Id	EndStation Name	Start Date	StartStation Id	StartStation Name
0	50754225	240	11834	2016-01-10 00:04:00	383.0	Frith Street, Soho	2016-01-10 00:00:00	18	Drury Lane, Covent Garden
1	50754226	300	9648	2016-01-10 00:05:00	719.0	Victoria Park Road, Hackney Central	2016-01-10 00:00:00	479	Pott Street, Bethnal Green
2	50754227	1200	10689	2016-01-10 00:20:00	272.0	Baylis Road, Waterloo	2016-01-10 00:00:00	425	Harrington Square 2, Camden Town

Figure 1.3: Sample data of each trip

As seen within the dataset, each row refers to one trip with information such as date, time, starting point, ending point and duration of ride. The dataset runs for 87 days from 01/06/2019 to 27/08/2019.

1.2.1 Data wrangling

You can refer to Part 1 of the Jupyter notebooks for the detailed description of the data wrangling methods. To summarize, the following steps were taken:

1. Check all columns are of the right type
2. Check for null values
3. Check values are consistent e.g. station IDs matched with station names, duration matched with difference in time
4. Check for any outliers

From step 1 and 2, we see that the date time values are recognized correctly and that there are no null values. However, we notice that there are some large outliers in the dataset and some of the station names and IDs are not matched due to discrepancies in spacing and punctuation. We also notice that some of the stations do not exist on the station dataset provided by the company, hence we remove these assuming errors in observations which removes $<2\%$ of our dataset. As for naming differences, we simply fix them for consistency and decide to keep the outliers as it most likely reflects the behaviour of not returning the cycles properly than an actual recording error.

1.2.2 Feature Engineering

This section is covered partly in Part 1 and 3 of the Jupyter notebooks. For the dataset to be provide more useful information for better prediction, we can add more features. In this case, we add the hour of day (0-24), type of day (weekday/weekend) and weather conditions as additional features. While the first feature is easily extracted from the dataset, the other 2 require a bit more work. The day of week (Mon-Sun) can be found using datetime functions but it doesn't tell us anything about public holidays. Here, we use the `holidays` package to also identify UK holidays.

As for weather information, we can pull historical data and forecasts from a weather API for the given dates. We get temperature, wind speed and a categorical variable for overall condition for every half hour or hour. Once again, we need to go through the data wrangling steps before we can merge this data with our feature dataset. The only issues are the outliers in wind speed that seem to be most likely an error in recording. Instead of just removing the data, we manage these by interpolating between available clean data. After cleaning the data, the final problem to tackle is the categorical weather data with 26 different categories. Most machine learning models will not perform well on categorical data, especially one with so many variants. This is dealt with by shrinking the categories to 4 main ones, namely good weather, ok weather, bad weather and very bad weather. We can then encode these categories into binary variables and merge this data with our main dataset.

Finally, we reduce our dataset to give daily aggregations. This is so that the model can predict next day demand based on previous days' data. We also add 3 more features: the number of bikes taken from each station the day before, the number of bikes docked at the station the day before and the 7 day rolling average of the duration of rides starting at the station. The possible Y variables to predict are number of rides starting at a station on a given day, the number of rides ending at a station on a given day and the average duration of rides starting at the station on a given day. The final features can be seen in 1.1

StartStation Id	Date	day_no	is_weekday	start_count_day_bf	end_count_day_bf	7d_rolling_dur	temperature	wind_speed	good_weather	ok_weather	bad_weather	very_bad_weather
	2019-06-09	6	0	19.0	9.0	12.701900	15.111111	11.777778	0.703704	0.296296	0.000000	0.0
	2019-06-10	0	1	27.0	17.0	12.746154	13.375000	17.750000	0.125000	0.375000	0.500000	0.0
	2019-06-11	1	1	8.0	5.0	13.381408	14.250000	11.300000	0.850000	0.150000	0.000000	0.0
1	2019-06-12	2	1	20.0	14.0	13.483942	12.903226	7.806452	0.741935	0.161290	0.096774	0.0
	2019-06-13	3	1	31.0	19.0	13.189248	14.095238	28.523810	0.619048	0.380952	0.000000	0.0
	2019-06-14	4	1	21.0	14.0	13.170451	16.550000	23.550000	1.000000	0.000000	0.000000	0.0
	2019-06-15	5	1	20.0	16.0	14.681730	17.681818	23.909091	1.000000	0.000000	0.000000	0.0

Table 1.1: Features for model

Chapter 2

Exploratory Analysis

2.1 Weather

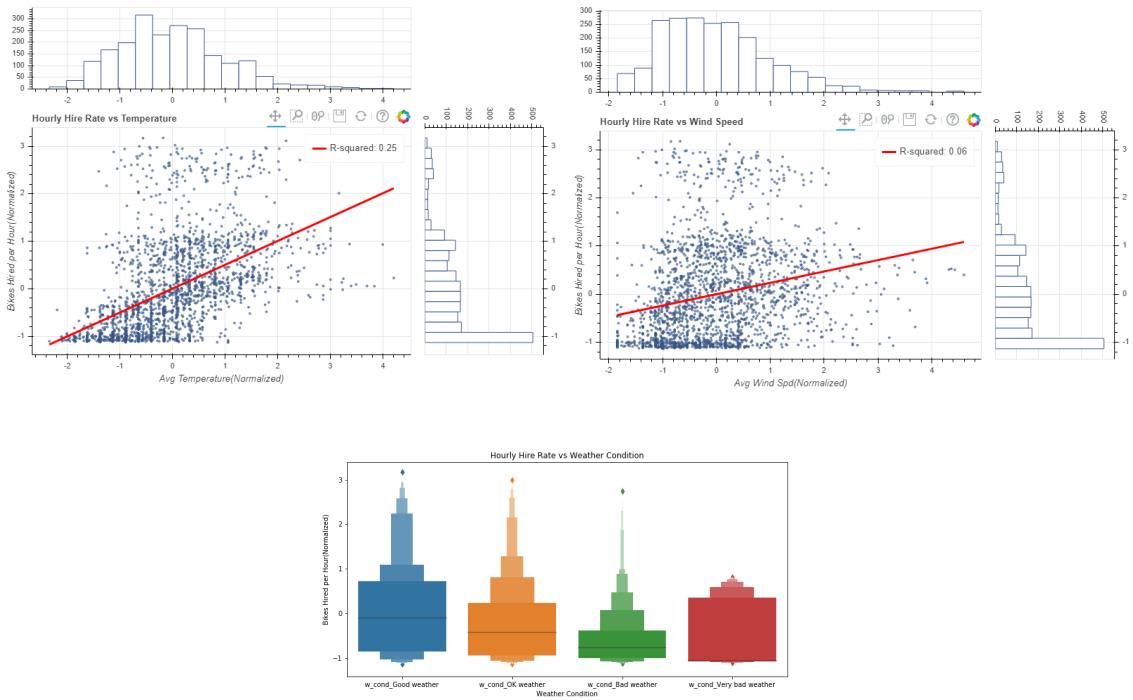

Figure 2.1: Relationship between weather conditions and number of rides

The relationship between temperature and ride count is significant and the number of rides increase with temperature. The relationship between wind speed and ride count is less significant but there still seems to be a weak positive correlation. As for the categorical weather conditions, it is clear that the ride count decreases significantly with worsening weather. There are some outliers that skew the data significantly but the median values clearly show a negative relationship.

2.2 Type of Day

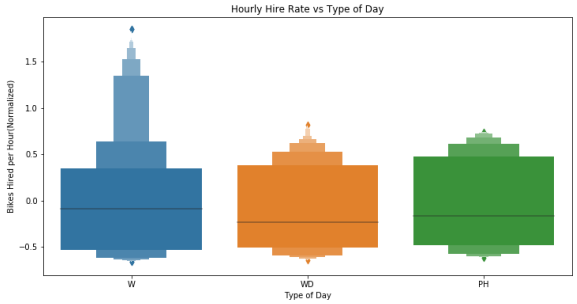

Figure 2.2: Relationship between type of day and number of rides

Weekdays seem to have higher frequency of bike hires and a greater spread of hourly hires.

2.3 Hour of Day

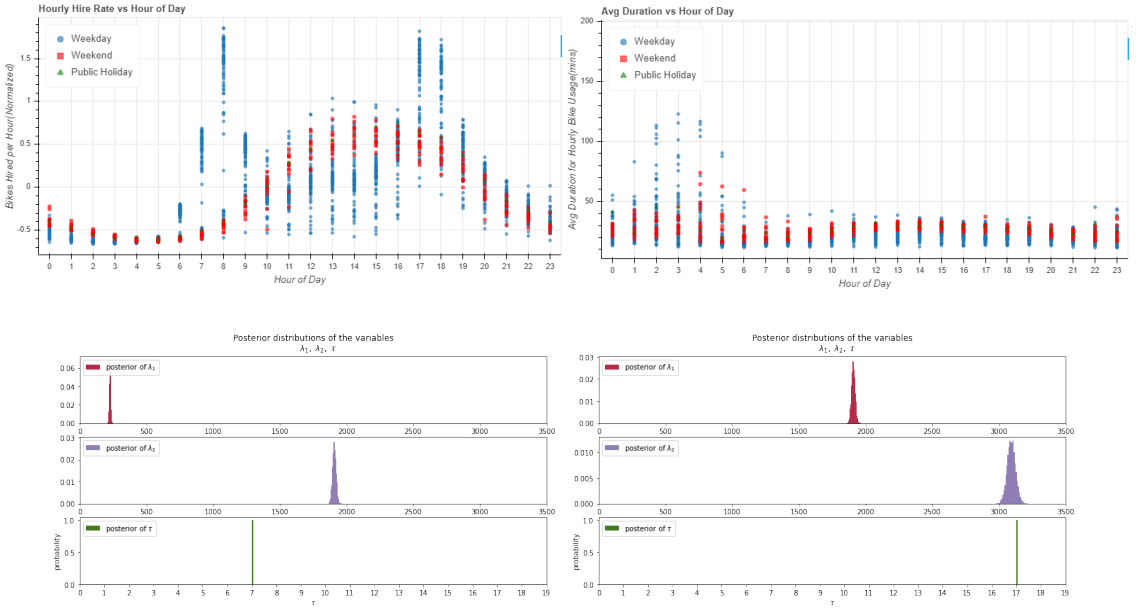

Figure 2.3: Relationship between hour of day and ride count and ride duration

The hour of day clearly has an influence on the number of bikes hires. We can see that there are higher peaks on weekdays around peak hours while weekends have a more normal distribution with a mean around 2pm. As for the duration of rides, they seem to be higher during night time when the frequency is at its lowest. We can also use bayesian inference to understand when during the day the hiring of bikes change. The analysis tells us that there is change in frequency of bike hires around 7am and 5pm which coincides with peak hours in London (when people leave for work and come back from work).

2.4 Ride count vs duration

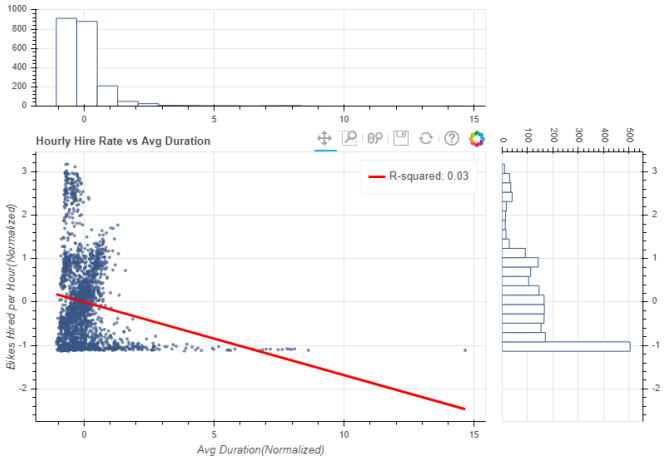

Figure 2.4: Relationship between ride count and duration of rides

We can also look to see if the frequency of bike hires is inversely proportional to the duration as observed in 2.3. There seems to be a very weak negative relationship which is not enough to support our initial observations.

Chapter 3

Machine Learning

3.1 Evaluating models

	Training Error	Test Error
Benchmark Model	24.27	26.65
Linear Regression	22.08	24.38
Gradient Boosting	18.46	23.54
XGBoost	20.87	23.53
AdaBoost	20.08	23.24

Table 3.1: Results from Machine Learning Models

We looked 5 different models in our analysis. You can refer to them in Part 5 of the Jupyter notebooks. The first model was the benchmark model which assumes the number of cycles hired in a given day is equals to the number of cycles hired on the previous day. The second model is a simple linear regression which gives some improvement in the performance. We then try three different boosting methods with AdaBoost providing the best improvement in performance.

Upon looking at the final results, we can see that the model does not predict the peak values very well. These peak values are generally from the same few stations and the model misses them as it hasn't learned the difference in location (i.e no features to distinguish different locations). We can address this issue by adding the station number as a feature or even by performing the analysis in separate groups, where each group is defined by a level of popularity.

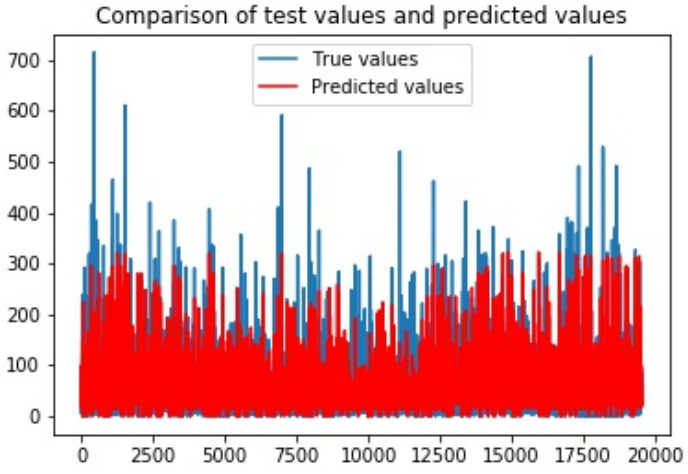

Figure 3.1: Results from AdaBoost Model

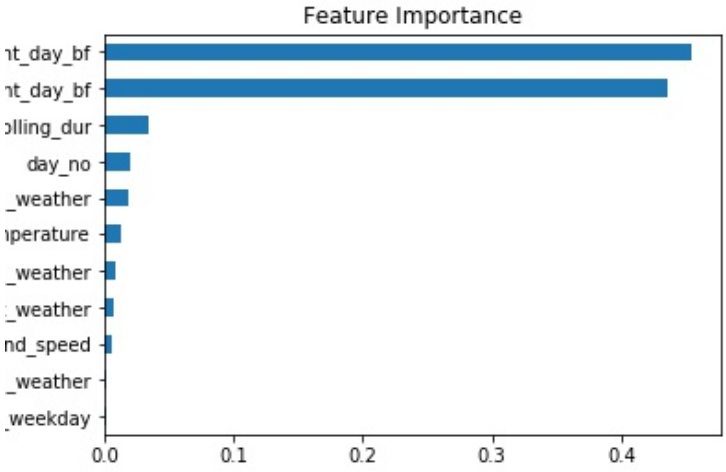

Figure 3.2: Feature Importances from AdaBoost Model

List of Figures

List of Tables